PCT/GB2003/003305.

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

RECD 08 OCT 2003
WIPO PCT

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

GB03/3305

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

I also certify that the attached copy of the request for grant of a Patent (Form 1/77) bears an amendment, effected by this office, following a request by the applicant and agreed to by the Comptroller-General.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed R. Mahoney

Dated 24 September 2003

Patents Form 1/77
(Rule 16)

THE PATENT OFFICE
D
26 JUL 2002
LONDON

The Patent Office

29JUL02 E736536-1 C87675
P01/7700 0.00-0217393.8

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference SMG/NAL1

2. Patent application number *(The Patent Office will fill in this part)* 0217393.8

26 JUL 2002

84335000 1

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

DR SARAH MARGARET GARDNER
30 BOLTON ROAD, WINDSOR, BERKSHIRE SL4

MR NATIVIDADE ALBERT LOBO
30 BOLTON ROAD, WINDSOR, BERKSHIRE SL4 3JT

Patents ADP number *(if you know it)*

If the applicant is a corporate body, give the country/state of its incorporation 30131133003

4. Title of the invention WIRELESS IDENTITY TRACING SYSTEM (WITS) FOR TRACING ANIMALS AND FOOD PRODUCTS

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

1/77
26/3
Pm.

Kilburn + Strode
20 Red Lion St
London
WC1R 4PJ.

~~30 BOLTON ROAD~~
~~WINDSOR~~
~~BERKSHIRE~~
~~SL4 3JN~~

Patents ADP number *(if you know it)* 125001 30131133003

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	16
Claim(s)	0
Abstract	0
Drawing(s)	0

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature Sarah M Gardner Date 25/7/02

N. A. Lobo 25/7/02

12. Name and daytime telephone number of person to contact in the United Kingdom

DR SARAH M GARDNER
01753 855504

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

Title of the Invention:

Wireless Identity Tracing System (WITS) for tracing animals and food products

Field and Background of the Invention

Tracking and tracing the contacts between animals or between food products or between animals and food products for the purpose of monitoring the transmission of disease.

Problem that the Invention Solves

Disease can spread through contact between animals or through contamination of food arising from contact between food and animal products. Presently it is not possible to maintain a record of the contacts that an individual animal makes with other individuals of the same and/or different species. Similarly, it is not possible to record the contacts between different food products, which may indicate possible sources of contamination between them. An object of this invention is to enable a record of contacts to be collected and maintained for individual animals and for individual food products.

In the wild, animals may come into contact with each other at points such as feeding or drinking areas, courtship territories or breeding grounds and migration routes. The contacts may be between animals of the same or different species including domestic livestock. Contacts between domestic and wild animals may be important in the spread of animal diseases such as the possible transmission of tuberculosis between badgers and cows. In the absence of direct observation or video recording, evidence of actual contact between the different species is difficult to obtain.

In modern livestock farming, animals can be moved between several different locations during their lifetimes. For example, breeding stock are normally reared on specialist livestock farms and then transferred to other farms to complete their productive lives. This transfer usually involves an auction market and at least one livestock dealer and two or more haulage contractors. At the end of their productive lives, the animal may pass through a cull stock market before being transferred to an abattoir or renderer. A similar or greater number of transfers may be recorded for animals reared for prime meat.

In the food industry, food products may pass through several locations before reaching their final point of sale. For example, the production of a joint of meat for sale in a supermarket will usually involve an abattoir, a butchering venue, a wrapping plant, a distribution outlet and a transport vehicle. At some or all of these venues, the joint may come into contact with joints or products from other animal carcasses and/or other food products.

What the Invention does

An object of this invention is to provide a record of the contacts that an individual animal has made with other animals of the same or of a different species or with individual food items. A further object of this invention is to provide a record of contacts made by an individual food item with other food items of the same or of a different type or with an animal.

Essential features of the Invention

The invention provides two types of electronic data collecting device, each comprising a radio receiver and transmitter, a processor for controlling the operation of the device and memory for storing information including a first unique identifier associated with the device.

Accordingly the first type of electronic data collecting device may be provided for an animal or a food item and is limited by its physical size, by its memory size and by its energy source. This device is named the *WITS Animal Product Device.* Similarly the second type of electronic data collecting device may be provided for facilities such as a building, a vehicle, food processing equipment, feeding troughs, marker posts and similar. This device is named the *WITS Facility Device* and normally includes a Geographical Positioning System (GPS) and may be linked to an external power source. It is anticipated that the WITS Facility Device will not be energy constrained and will possess more power, more memory and a stronger signal than the WITS Animal Product Device and will be able to transmit over a greater range.

For both devices, the processor is arranged to receive, by means of a radio receiver, a wakeup call at all times. The wakeup call consists of the unique identifier belonging to another electronic data collecting device or to an electronic reader. All wakeup calls are evaluated by the processor and either stored in the memory or discarded. The procedure determining whether a wakeup call is stored or discarded is outlined in the *Process Specification* below and is described further in the attached simulation model. Preferably the date on which the wakeup call was received is also stored together with the unique identifier of the transmitting device, said date representing information on when the contact was made.

For both devices, the processor is also arranged to transmit its First Unique Identifier by means of a radio transmitter. To conserve energy, the radio transmitter is normally in "deep sleep" and is switched on after the processor has received a wakeup call according to a pre-determined time interval or a wakeup call from another electronic data collecting device or a wakeup call from an electronic reader. The process by which the WITS Animal Product Device and the WITS Facility Device transmits its First Unique Identifier in response to a wakeup call is outlined in the ***Process specification*** section below and described further in the attached model. When transmitting, the WITS Animal Product Device and the WITS Facility Device merely transmit their own First Unique Identifier. They do not establish a communication link with other electronic data collecting devices.

Both the WITS Animal Product Device and the WITS Facility Device are designed to respond to a wakeup call from an electronic *WITS Reader* and to a request from the said reader to download their records to the reader. The principal function of the WITS reader is to gather the identities stored in WITS Animal Product Devices and WITS Facility Devices. The reader is therefore able to send a wakeup call to the said devices and to communicate with a standard PC or similar, so that the information from the WITS devices can be processed.

A description of the behaviour of the WITS devices

The behaviour of the WITS Animal Product Device and the WITS Facility Device is now described using CSP notation. In the following description, N, N1 *etc.* always denotes the identity of the device. The variable n, n1 *etc.* describes the identity of any other device in the vicinity. The events used in the ***Process Specification*** below are described in Table 1.

Table 1. *The alphabet of events occurring in the Process Specification*

EVENT	DESCRIPTION OF THE EVENT
Receive[n]	Term used to describe the event whereby a WITS Animal Product Device or a WITS Facility Device identifies that it has received a wakeup call and a Unique Identifier from another collecting device [n] making the transmission. To safeguard against fraud, the signal strength of the wakeup call will also be recorded using a flag that indicates whether the signal exceeds a given strength. If two devices were co-located on or in the same animal or food product, the transmission received would exceed the expected strength by a significant amount. By examining the store, these offending devices could be monitored.
Transmit[N]	Term used to describe the event whereby a WITS Animal Product Device [N] or a WITS Facility Device [N] broadcasts its First Unique Identifier. This event occurs in response to a Receive[n] event or a TimeToTransmit[N] event. The Transmit[N] event always includes a wakeup call. Note that each Device [N] can only broadcast its own First Unique Identifier, it does not broadcast the identities of any other devices [n] that it has stored. When the Transmit[N] event is in response to a Receive[n] event, the device waits for a random period of between 0 and MaxWait seconds. The duration of MaxWait is to be determined from consideration of the WITS system. The Transmit[N] event is an energy expensive operation and the WITS system restricts the number of times that the transmitter in the WITS Animal Product Device is turned on. Thus the WITS Animal Product Device transmits infrequently, say once every 5 minutes, and the WITS Facility Device transmits frequently, say once every second, and uses a stronger signal than the WITS Animal Product Device.
TimeToTransmit[N]	The TimeToTransmit[N] event is a function of the WITS system clock and small variations in the frequency will cause different WITS data collecting devices to actually fire at different times. This is the randomisation procedure used in Bluetooth.
WhatIsInRecentStore[n]	Event that occurs when a WITS Reader is interrogating a WITS Animal Product Device [n] or a WITS Facility Device [n].
WhatIsInStore[n]	Event that occurs when a WITS Reader is interrogating a

	WITS Animal Product Device [n] or a WITS Facility Device [n].
ReadOutRecentStore[N]	Term used to describe the event whereby a WITS Animal Product Device [N] or WITS Facility Device[N] transmits the contents of its Recent Store to a WITS Reader. Note that the devices can only transmit the data in their own stores.
ReadOutStore[N]	Term used to describe the event whereby a WITS Animal Product Device [N] or WITS Facility Device[N] transmits the contents of its Entire Store to a WITS Reader.
PutInStore[n]	This event is an abstraction of the process involved in storing the data. The Store may be organised in the form of several databases and the WITS data collecting devices may use a two step process whereby, for example, the devices store data temporarily in a short-term memory before moving these data or a selection of the data into a more permanent place.
NotInStore[n]	This event may occur after just a small search or after a more global search of the Store is made. The size of the search process will be decided after consultation with the customer.
FoundInStore[n]	This event occurs if NotInStore[n] does not occur

Process Specification

In this specification the term Collector refers to a WITS Animal Product Device or to a WITS Facility device. *Algorithm 1* sets out the process whereby a WITS Animal Product Device [N]or a WITS Facility Device [N] broadcasts its First Unique Identifier, stores the Unique Identities that it receives from other collecting devices and downloads its Store or RecentStore to a WITS Reader.

Algorithm 1:

Collector[N] = μX[N].((Receive[n] → (NotInStore[n]→ PutInStore[n]→
Transmit[N] | FoundInStore[n]→ X[N]) |
TimeToTransmit[N] →Transmit[N] → X[N] |
WhatIsInRecentStore[n] → (X[N] | ReadOutRecentStore[N] → X[N]) |
WhatIsInStore[n] →(X[N] | ReadOutStore[N] → X[N]))

Algorithm 2 sets out the process for a WITS Reader with identity N to read the store of a sheep that it is looking at.

Algorithm 2: Let the sheep identity be n1, then

ReadRecentStore[N,N1] =μX[N,N1].(ReadRecentStore[N1]→ □)

Algorithm 3 sets out the process whereby a WITS Reader can read any sheep store.

Algorithm 3:

ReadRecentStore[N] = μX[N].(Transmit[N]→ Receive[n1] →
ReadRecentStore[N,n1]→ □)

Simulation Model of the WITS system

The model below describes the process of information exchange between a group of WITS electronic data collecting devices within transmission range of each other. The WITS Animal Product Device transmits over a small distance, say 10 m, which has yet to be fully specified. This WITS Facility Device transmits over a larger distance which has also to be fully specified. The model does not describe the process of information exchange between WITS electronic data collecting devices and a WITS reader. An appropriate standard data exchange mechanism would be used for this process.

The collector is defined below and represents either a WITS Animal Product Device or a WITS Facility Device. By varying the TransmitPeriod, the time interval between two normal transmission, can be varied. WITS Facility Devices do not have a power problem and can have a short transmit period. Again, the memory size can be varied so that WITS Facility Devices can be built with plenty of memory. Clock crystals are known to vary with temperature and from one device to another. Therefore, the time on the clock in every device will show considerable variation after some time.

```
CollectorDevice := Collector[DeviceIdentity[name],
  OperatingParameters[{TransmitPeriod[tp], MemorySize[mem]}],
  Components[{ClockFrequency[f]}],
  FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[tp]}],
  State[{DeviceClockTime[dt], LastTransmission[lt],
    NextTransmission[nt], OperationalMode[mode], ContactsRecord[rec]}],
  SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]
```

Transmitting Rate is 12.8 KBits/s.

```
Minute = 60 s
```

60 s

```
Hz = 1 / s
```

$$\frac{1}{s}$$

```
NumberInFlock = 4
```

```
4
```

```
NominalFrequency := 10^4 Hz
```

We assume that the clock frequency is accurate to one part in a million. The actual component may have worse accuracy. The cheaper the crystal the less accurate the frequency is. We assume a transmit period of 10 minutes.

```
Flock = Table[
   CollectorDevice /. {name :→ i, f :→ NominalFrequency ( 1 + 10^-6 (Random[] - 0.5) ),
     rec :→ {}, tp :→ 10 Minute , mem :→ 10}, {i, 1, NumberInFlock}];
```

We assume that all the clocks are synchromised at t= 0.

We set all the clocks to 0 and set all of them to have last tansmitted at 0.The numbers in the NextTransmit are the actual time. A ten minute interval will cause the NextTransmit time to be 600s. However, because the clocks do not beat at the correct frequency, some run faster and may appear to clock 600s after 599.5 s say. The model records the actual time of the transmisson to be 599.5s. In the actual device it would be recorded as 600s.

```
InitialiseTime[Flock_] := Flock = (Flock /. {dt :→ 0, lt :→ 0})
```

```
InitialiseTime[Flock];
```

```
Flock[[1]]

Collector[DeviceIdentity[1],
 OperatingParameters[{TransmitPeriod[600 s], MemorySize[10]}],
 Components[{ClockFrequency[10000./s]}],
 FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
 State[{DeviceClockTime[0], LastTransmission[0],
   NextTransmission[nt], OperationalMode[mode], ContactsRecord[{}]}],
 SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]
```

NextScheduledTransmitTime calculates the next time to transmit given that transmit period and the actual time.

```
NextScheduledTransmitTime[ClockFrequency[f_], TransmitPeriod[tp_], ActualTime_] :=
 Module[{x1, x2, x3, x4, x5, x6, x7, x8},
      x1 = ActualTime f;
      x2 = tp NominalFrequency;
      x3 = Ceiling[x1 / x2] x2;
      x3/f]
```

CalculateNextTransmitTime decides to set the transmit time to be at least the transmit period more than the next scheduled transmit time

```
CalculateNextTransmitTime[ClockFrequency[f_], TransmitPeriod[tp_],
  LastTransmission[lt_], NextScheduledTransmitTime_] :=
    Module[{x1, x2, x3, x4},
  x1 = tp NominalFrequency/f;
  x2 = lt + x1;
  Max[x2 / s, NextScheduledTransmitTime / s] s
 ]


SetNextTransmission[Sheep_, ActualTime_] :=
     Module[{x1, x2, x3, x4, x1a, x2a, x3a, x4a, x5, x6, x7, x8},
  x1 = Position[Sheep, NextTransmission[_]];
  x2 = Position[Sheep, ClockFrequency[_]];
  x3 = Position[Sheep, TransmitPeriod[_]];
  x4 = Position[Sheep, LastTransmission[_]];
  x2a = Extract[Sheep, x2][[1]];
  x3a = Extract[Sheep, x3][[1]];
  x4a = Extract[Sheep, x4][[1]];
  x5 = NextScheduledTransmitTime[x2a, x3a, ActualTime];
  x6 = CalculateNextTransmitTime[x2a, x3a, x4a, x5];
   Sheep /. NextTransmission[x_] → NextTransmission[x6] ]

InitialiseNextTransmission[StartTime_, Flock_] :=
 Module[{x1, s2, s3, s4},
  Flock = Map[SetNextTransmission[#, StartTime] &, Flock ];
 ]
```

```
InitialiseNextTransmission[0 s, Flock]
```

Now we find the next transmission to execute

```
FindNextTransmissionToExecute[Flock_] :=
  Module[{x1, x2, x3, x4, x5},
   x1 = Position[Flock, NextTransmission[_]];
   x2 = (Extract[Flock, x1] /. NextTransmission[x_] :> (x / s));
   x3 = Min[x2];
   x4 = Position[x2, x3]
  ];

FindNextTransmissionToExecute[Flock]
```

```
{1}
```

```
GenerateTransmission[Sheep_] :=
 Module[{x1, x2, x3},
  x1 = Position[Sheep, DeviceIdentity[_]];
  x2 = Position[Sheep, NextTransmission[_]];
  x3 = Position[Sheep, CoOrdsOfPosition[_]];
  x1a = Extract[Sheep, x1][[1, 1]];
  x2a = Extract[Sheep, x2][[1, 1]];
  x3a = Extract[Sheep, x3][[1, 1]];
  {x1a, x2a, x3a}]
```

```
GenerateTransmission[Flock[[4]]]
```

```
{4, 600. s, {x, y, z}}
```

The state change required on transmission is quite simple. The contents of the next state are transfered into the location used tostore the last transmission. The time to make the next transmission is set according to the transmit period

```
ChangeStateOnTransmission[Sheep_] :=
 Module[{x1, x2, x3, x4, x5, x6, x7},
  x1 = Position[Sheep, NextTransmission[_]];
  x2 = Position[Sheep, LastTransmission[_]];
  x6 = Position[Sheep, ClockFrequency[_]];
  x7 = Position[Sheep, TransmitPeriod[_]];
  x1a = Extract[Sheep, x1][[1, 1]];
  x2a = Extract[Sheep, x2][[1, 1]];
  x6a = Extract[Sheep, x6][[1, 1]];
  x7a = Extract[Sheep, x7][[1, 1]];
  x3 = (x1a /. NextTransmission[x_] ↦ x);
  x4 = (Sheep /. LastTransmission[x_] ↦ LastTransmission[x3]) ;
  x5 = (x3 + x7a NominalFrequency / x6a);
  x8 = (x4 /. NextTransmission[x_] ↦ NextTransmission[x5]) ]
```

```
TransmitPeriod[600 s]
```

```
Flock[[1]]

Collector[DeviceIdentity[1],
 OperatingParameters[{TransmitPeriod[600 s], MemorySize[10]}],
 Components[{ClockFrequency[10000./s]}],
 FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
 State[{DeviceClockTime[0], LastTransmission[0],
   NextTransmission[600. s], OperationalMode[mode], ContactsRecord[{}]}],
 SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]
```

Update the record and state describes what happens when the sheep receives the data. It may need to change the next transmission time if it accepts the record.

```
UpdateTheRecordAndState[Sheep_, ReceivedData_] :=
 Module[{x1, x2, x3, x4, x5, x6},
  x1 = Position[Sheep, ContactsRecord[_]];
  x2 = Position[Sheep, MemorySize[_]];
  x3 = Position[Sheep, NextTransmission[_]];
  x1a = Extract[Sheep, x1][[1, 1]];
  x2a = Extract[Sheep, x2][[1, 1]];
  x3a = Extract[Sheep, x3][[1, 1]];
  {x5, x4} = ContactRecordUpdate[x1a, x2a, x3a, ReceivedData];
  x6 = (Sheep /. NextTransmission[x_] :> x4);
  x7 = (x6 /. ContactsRecord[x_] :> x5)
 ]
```

We expect ContactRecordUpdate to return with the new transmission time and the new record. These may be the same as the old ones in both cases.

```
MinTimeSeparation = 600.0 s;

ContactRecordUpdate[cr_, ms_, nt_, ReceivedData_] :=
 Module[{x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12},
  x4 = ReceivedData[[1]];
  x5 = ReceivedData[[2]]/s;
  x6 = ReceivedData[[3]];
  x7 = Select[cr, #[[1]] == x4 &];
  x8 = (Map[#[[2]]/s &, x7]);
  x9 = Map[((MinTimeSeparation/s) < x5 - #) &, x8];
  x10 = Apply[And, x9];
  x11 = AddToContactRecord[x10][cr, ms, ReceivedData, nt] ]

AddToContactRecord[False][cr_, ms_, Record_, nt_] :=
 {ContactsRecord[cr], NextTransmission[nt]}
```

```
MaxDelayOfResponse := 1.0

AddToContactRecord[True][cr_, ms_, Record_, nt_] :=
 Module[{x1, x2, x3, x4, x5, x6, x7, x1a, x2a, x3a},
  x1a = Record // #[[2, 1]] &;
  x1 = (Length[cr] < ms);
  x2[False] := RemoveEntries[{cr, x1a}];
  x2[True] := cr;
  x3 = x2[x1];
  x4 = Join[{Record}, x3];
  x5 = Record[[2]] / s;
  x6 = x5 + MaxDelayOfResponse (0.5 + Random[]);
  {ContactsRecord[x4], NextTransmission[x6 s]}]

RemoveEntries[{cr_, currenttime_}] :=
 Module[{x1, x2, x3, x4, x5},
  x1 = Sort[cr];
  x2 = Split[x1, (#1[[1]] = #2[[1]]) &];
  x3 = Select[x2, (Length[#] > 2) &];
  x4 = Complement[x2, x3];
  x5[{}] := (Print["There are at most 2 rcords Per Sheep"]; x4);
  x5[x_] /; x ≠ {} :=
   (Map[PruneEntriesOfEachSheep[currenttime], x3] // Join[x4, #] &);
  x6 = x5[x3];
  x7 = Flatten[x6, 1]
 ]


PruneEntriesOfEachSheep[CurrentTime_][SingleSheepsRecord_] :=
 Module[{x1, x2, x3, x4},
  x1 = Map[#[[2, 1]] &, SingleSheepsRecord];
  x2 = CurrentTime - x1;
  x3 = Partition[x2, 2, 1];
  x4 = Map[#[[1]] / #[[2]] &, x3];
  x5 = Position[x4, Min[x4]] // Flatten // #[[1]] &;
  x6 = Drop[SingleSheepsRecord, {x5 + 1}]
 ]

Clear[BasicLoop]
```

```
BasicLoop[Flock_] :=
 Module[{x1, x2, x3, x4, x5, x6},
  x1 = FindNextTransmissionToExecute[Flock];
  x2 = x1[[1, 1]];
  x3 = GenerateTransmission[Flock[[x2]]];
  x4 = ChangeStateOnTransmission[Flock[[x2]]];
  x5 = Drop[Flock, {x2}];
  x6 = Map[UpdateTheRecordAndState[#, x3] &, x5];
  Flock = Insert[x6, x4, x2]]

BasicLoop[Flock]

Collector[DeviceIdentity[2],
 OperatingParameters[{TransmitPeriod[600 s], MemorySize[10]}],
 Components[{ClockFrequency[10000./s]}],
 FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
 State[{DeviceClockTime[0], LastTransmission[600. s],
   NextTransmission[600. s], OperationalMode[mode], ContactsRecord[{}]}],
 SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]

ChangeStateOnTransmission[Flock[[4]]]

Collector[DeviceIdentity[4],
 OperatingParameters[{TransmitPeriod[600 s], MemorySize[mem]}],
 Components[{ClockFrequency[10000./s]}],
 FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
 State[{DeviceClockTime[0], LastTransmission[600. s],
   NextTransmission[600. s], OperationalMode[mode], ContactsRecord[{}]}],
 SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]

tom = InitialiseNextTransmission[Flock]

Flock = InitialiseTime[Flock];

ReceiverProg[2 + 2]

ReceiverProg[4]
```

An Example of the Ouput from the Simulation Model of the WITS system

```
NumberInFlock = 10
```

```
10
```

```
NominalFrequency := 10^4 Hz
```

```
Flock = Table[
   CollectorDevice /. {name :> i, f :> NominalFrequency ( 1 + 10^-6 (Random[] - 0.5) ),
     rec :> {}, tp :> 10 Minute , mem :> 20}, {i, 1, NumberInFlock}];
```

```
InitialiseTime[Flock];
```

```
InitialiseNextTransmission[0 s, Flock]
```

```
Flock[[1]]
```

```
Collector[DeviceIdentity[1],
 OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
 Components[{ClockFrequency[10000./s]}],
 FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
 State[{DeviceClockTime[0], LastTransmission[0],
   NextTransmission[600. s], OperationalMode[mode], ContactsRecord[{}]}],
 SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]
```

```
? BasicLoop
```

```
Global`BasicLoop
```

```
BasicLoop[Flock_] := Module[{x1, x2, x3, x4, x5, x6}, x1 = FindNextTransmissionToExecute[Flock];
  x2 = x1[[1, 1]]; x3 = GenerateTransmission[Flock[[x2]]]; x4 = ChangeStateOnTransmission[Flock[[x2]]];
  x5 = Drop[Flock, {x2}]; x6 = (UpdateTheRecordAndState[#1, x3] &) /@ x5; Flock = Insert[x6, x4, x2]]
```

```
Table[BasicLoop[Flock], {i, 1, 100}] // Last
```

```
{Collector[DeviceIdentity[1],
  OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
  Components[{ClockFrequency[10000./s]}],
  FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
  State[{DeviceClockTime[0], LastTransmission[1821.75 s],
    NextTransmission[2423.51 s], OperationalMode[mode],
    ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
      {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
      {4, 600.909 s, {x, y, z}}, {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}},
      {6, 2420.99 s, {x, y, z}}, {7, 602.822 s, {x, y, z}}, {7, 1815.36 s, {x, y, z}},
      {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}}, {10, 607.326 s, {x, y, z}},
      {10, 1818.15 s, {x, y, z}}, {5, 603.432 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
      {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
  SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
  DeviceIdentity[2],
  OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
```

```
Components[{ClockFrequency[10000./s]}],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[1821.1 s], NextTransmission[2423.38 s],
  OperationalMode[mode], ContactsRecord[{{7, 2422.84 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {3, 600. s, {x, y, z}},
    {3, 1814.66 s, {x, y, z}}, {4, 600.909 s, {x, y, z}}, {4, 2420.36 s, {x, y, z}},
    {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}}, {7, 602.822 s, {x, y, z}},
    {7, 1815.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {5, 603.432 s, {x, y, z}},
    {5, 2422.26 s, {x, y, z}}, {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[3],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[1821.79 s],
  NextTransmission[2423.47 s], OperationalMode[mode],
  ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {1, 605.06 s, {x, y, z}},
    {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}}, {2, 1813.53 s, {x, y, z}},
    {4, 600.909 s, {x, y, z}}, {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}},
    {6, 2420.99 s, {x, y, z}}, {7, 602.822 s, {x, y, z}}, {7, 1815.36 s, {x, y, z}},
    {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}}, {10, 607.326 s, {x, y, z}},
    {10, 1818.15 s, {x, y, z}}, {5, 603.432 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
    {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[4],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[2422.82 s],
  NextTransmission[2424.03 s], OperationalMode[mode],
  ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
    {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
    {5, 603.432 s, {x, y, z}}, {5, 1814.08 s, {x, y, z}}, {7, 602.822 s, {x, y, z}},
    {7, 1815.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {6, 606.258 s, {x, y, z}},
    {6, 2420.99 s, {x, y, z}}, {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[5],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[2422.26 s],
  NextTransmission[2423.84 s], OperationalMode[mode],
  ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
    {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
    {7, 602.822 s, {x, y, z}}, {7, 1815.36 s, {x, y, z}}, {8, 604.122 s, {x, y, z}},
    {8, 1817.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
```

```
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
    {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}}}]]]],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[6],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}]],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[2420.99 s],
  NextTransmission[2423.62 s], OperationalMode[mode],
  ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
    {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
    {5, 603.432 s, {x, y, z}}, {5, 1814.08 s, {x, y, z}}, {7, 602.822 s, {x, y, z}},
    {7, 1815.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
    {4, 2420.36 s, {x, y, z}}, {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]]]],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[7],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}]],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[2422.84 s],
  NextTransmission[3022.84 s], OperationalMode[mode],
  ContactsRecord[{{5, 2422.26 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
    {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
    {5, 603.432 s, {x, y, z}}, {5, 1814.08 s, {x, y, z}}, {8, 604.122 s, {x, y, z}},
    {8, 1817.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
    {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}}}]]]],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[8],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}]],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[2421.55 s],
  NextTransmission[2424.04 s], OperationalMode[mode],
  ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
    {1, 605.06 s, {x, y, z}}, {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}},
    {2, 1813.53 s, {x, y, z}}, {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}},
    {5, 603.432 s, {x, y, z}}, {5, 1814.08 s, {x, y, z}}, {7, 602.822 s, {x, y, z}},
    {7, 1815.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}}, {9, 1815.92 s, {x, y, z}},
    {10, 607.326 s, {x, y, z}}, {10, 1818.15 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
    {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}}}]]]],
SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
DeviceIdentity[9],
OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
Components[{ClockFrequency[10000./s]}]],
FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
State[{DeviceClockTime[0], LastTransmission[1821.08 s],
```

```
    NextTransmission[2423.82 s], OperationalMode[mode],
    ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {1, 605.06 s, {x, y, z}},
      {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}}, {2, 1813.53 s, {x, y, z}},
      {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
      {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}},
      {7, 602.822 s, {x, y, z}}, {7, 1815.36 s, {x, y, z}}, {10, 607.326 s, {x, y, z}},
      {10, 1818.15 s, {x, y, z}}, {5, 603.432 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
      {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
  SimulationState[{CoOrdsOfPosition[{x, y, z}]}]], Collector[
  DeviceIdentity[10],
  OperatingParameters[{TransmitPeriod[600 s], MemorySize[20]}],
  Components[{ClockFrequency[10000./s]}]],
  FunctionsOnState[{ReceiverProgram[rp], TransmitterProgram[600 s]}],
  State[{DeviceClockTime[0], LastTransmission[1821.49 s],
    NextTransmission[2424.12 s], OperationalMode[mode],
    ContactsRecord[{{7, 2422.84 s, {x, y, z}}, {1, 605.06 s, {x, y, z}},
      {1, 1816.47 s, {x, y, z}}, {2, 601.455 s, {x, y, z}}, {2, 1813.53 s, {x, y, z}},
      {3, 600. s, {x, y, z}}, {3, 1814.66 s, {x, y, z}}, {4, 600.909 s, {x, y, z}},
      {4, 2420.36 s, {x, y, z}}, {6, 606.258 s, {x, y, z}}, {6, 2420.99 s, {x, y, z}},
      {7, 602.822 s, {x, y, z}}, {7, 1815.36 s, {x, y, z}}, {9, 602.253 s, {x, y, z}},
      {9, 1815.92 s, {x, y, z}}, {5, 603.432 s, {x, y, z}}, {5, 2422.26 s, {x, y, z}},
      {8, 604.122 s, {x, y, z}}, {8, 2421.55 s, {x, y, z}}}]}],
  SimulationState[{CoOrdsOfPosition[{x, y, z}]}]]}
```